TASS USER MANUAL

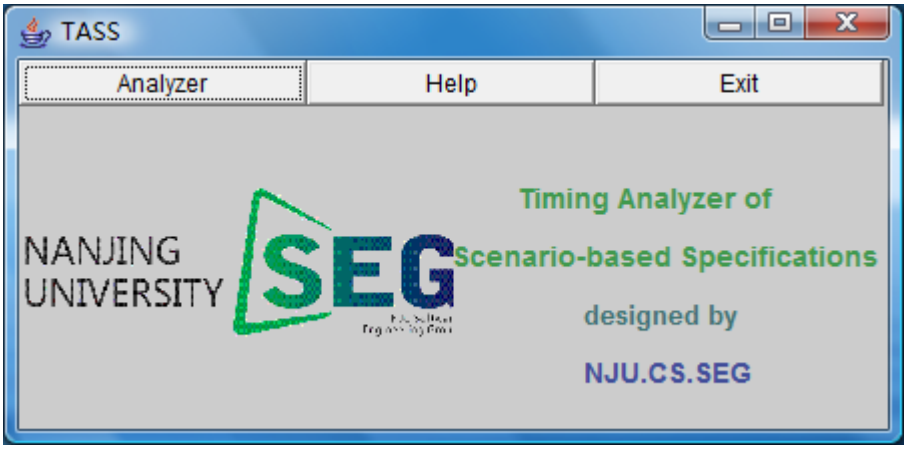

Table of Contents

1. Preface

This manual provides an introduction to TASS (**T**iming **A**nalyzer of **S**cenario-based **S**pecifications). As the name indicates, TASS is a timing analyzer for Scenario-Based specifications expressed by UML2.0 interaction models. TASS can be used for *reachability analysis*, *constraint conformance analysis*, and *bounded delay analysis* of scenario-based specifications. It is composed of three analyzers: *path-oriented timing analyzer*, *bounded timing analyzer* and *timing analyzer*.

2. Resources

- TASS is a freeware for non-commercial use.

3. Feedback

If you have any problems about TASS or this document, send email to panmx@seg.nju.edu.cn.

4. Write Scenario-Based Specifications

Scenario-based specifications such as UML2.0 interaction models offer an intuitive and visual way of describing design requirements, and are playing an increasingly important role in design of software systems. TASS uses Scenario-Based Specifications (SBSs) expressed by UML2.0 interaction overview diagrams whose nodes are sequence diagrams (SDs). While a single SD describes exactly one scenario, a SBS can describe multiple scenarios and complete system specifications.

We introduce more general and expressive timing constraints in TASS, and gives an approach to efficient timing analyze SBSs. Based on linear programming, TASS solves the reachability analysis, the constraint conformance analysis, and the bounded delay analysis problems.

Since UML is an OMG's standard modeling language and Rational Rose Enterprise (Rose for short) is one of the frequently used modeling tools in industry, we'll make use of UML diagrams and Rose to design the scenario-based specifications.

4.1 Sequence Diagrams

We just use a simplified version of UML sequence diagrams, which describes exactly one scenario without any alternative and loop. A SD considered here has two dimensions: the vertical dimension represents time, and the horizontal dimension represents different objects. Each object is assigned a col-

umn, and the messages are shown as horizontal, labeled arrows. We ignore the class concept in SD, so all the objects are “dummy objects” that do not specify classes which they belong to explicitly.

Steps of building a SD in Rose:

1. Before we create a new SD in Rose, we first need to create a new model.
2. A model has several views that describes a software system. Here we just need the “Logic View”. Add a new SD to Logic View, and name it.
3. Add objects and messages to the diagram. You need not to specify the class which objects belong to, just naming the objects and the messages is enough.

After accomplishing the steps, you should get a diagram similar to Figure 1.

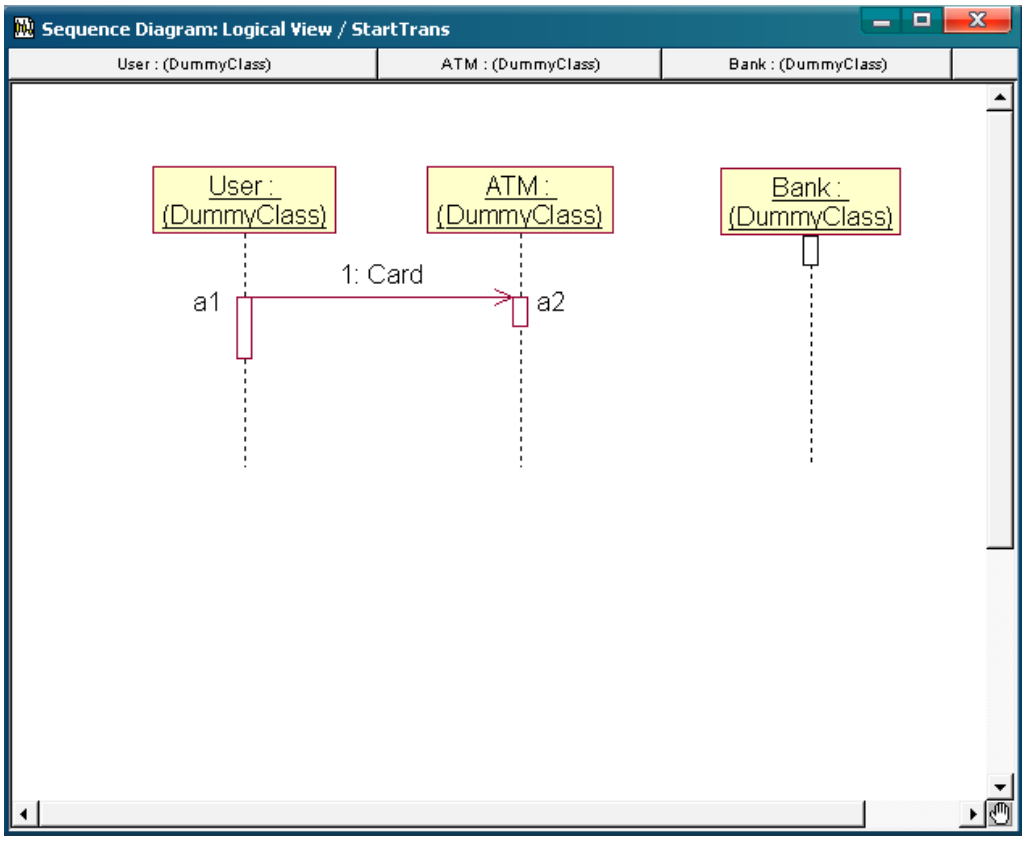

Figure 1 A sequence diagram in Rose

4.2 Scenario-Based Specifications

We use UML statechart’s constructs to represent a scenario-based specification. In detail, we use states in statechart to represent the nodes, and transitions in statechart as flow of control. We also make

use of the start state and end state, for start node and end node. Note that the name appears in the state must be the same as the SD name which is associated with the diagram.

Steps of building a statechart in Rose:

1. Add a new statechart to Logic View, and name it. Make sure there is only one statechart for a model.
2. Add states and transitions to the statechart. Be sure that each state name is just the same as the SD name which it maps to. Note that if a state appears in the diagram for more than one time and represents to the same SD scenario, renaming these states is necessary. And because one state can only be mapped to one SD, the user needs to create new sequence diagrams for these new states. But repeating all these work is not necessary, since these scenarios are the same except the diagrams' names are different, one only needs to copy all the diagram items and name the diagram differently.

After accomplishing the steps, you should get a diagram similar to Figure 2.

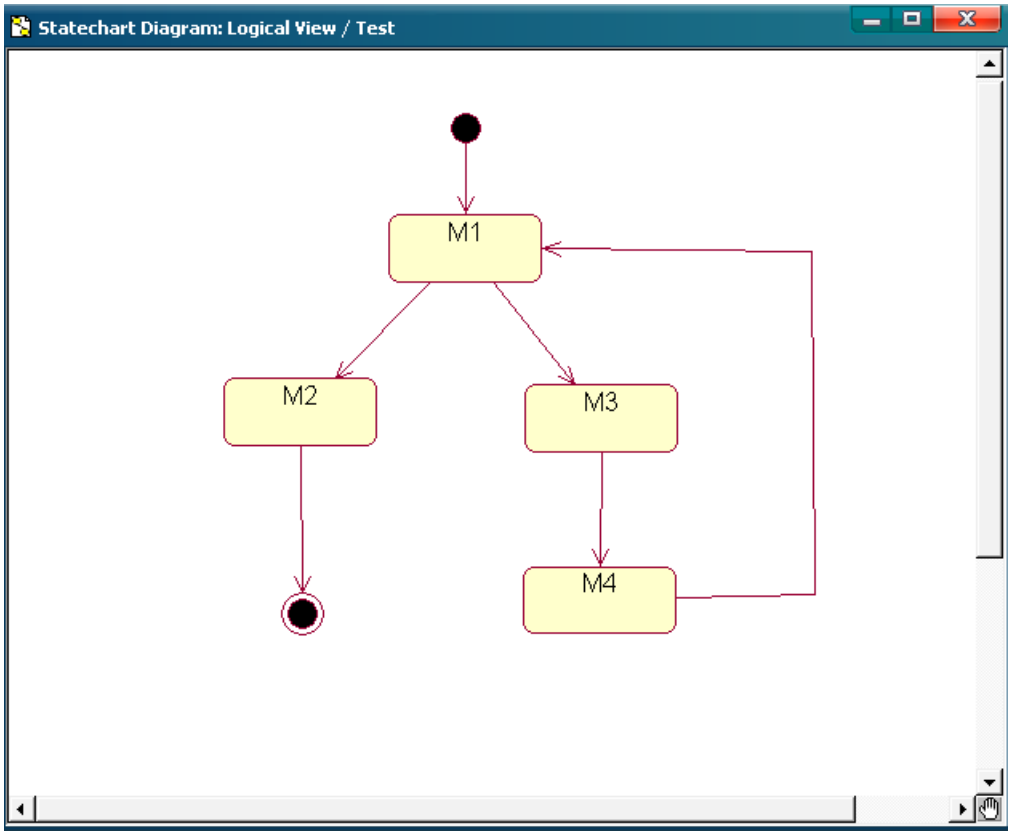

Figure 2 A model in Rose

4.3 Export the Model to a XML File

XMI Add-in is used to export the model to a XML file. You should first install it. The download link can be found in the appendix. Once you've installed it successfully, there will be a new menu item named as "UML 1.3 XMI add-in" in the Tools menu. Before you export the model, you should save the model. Choose Tools→UML 1.3 XMI add-in→UML 1.3 XMI Export.

The option dialog is as Figure 3. Make sure the options of "Export Diagrams", "XMI 1.1" for XMI Format and "Model" for Output File Basis are chosen. Other options depend on your requirements and TASS can handle the differences of the exporting files.

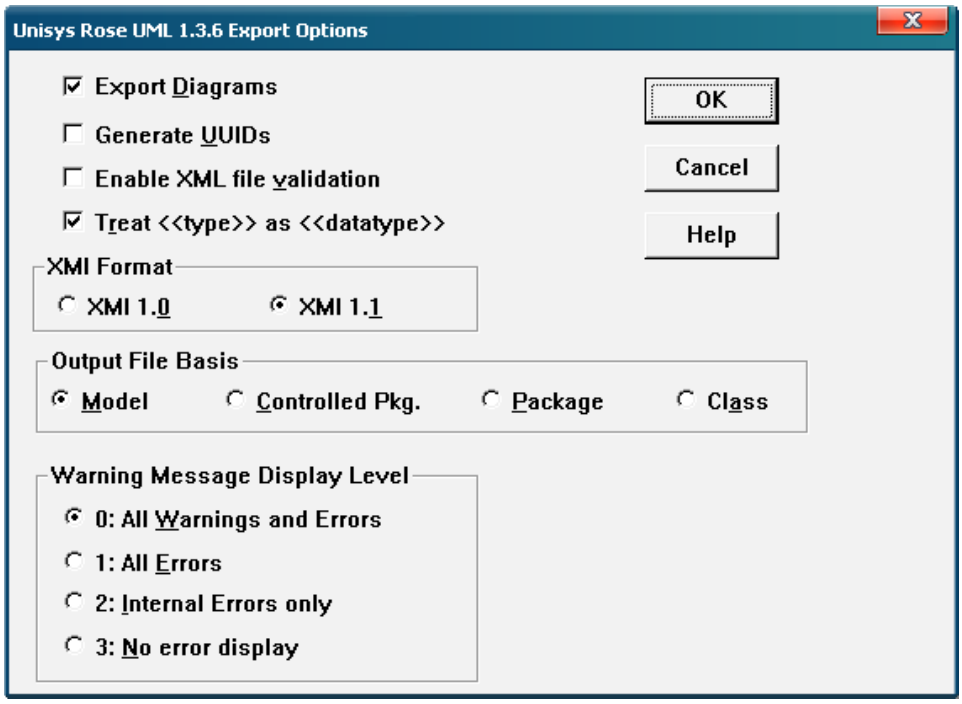

Figure 3 XMI add-in options

4.4 Introduction to TASS GUI

To grant maximum degree of comfort, the TASS GUI is partitioned into six components (cf. Figure 4). Component (1) displays all the elements in a scenario-based specification including the sequence diagrams as well as the communication time between objects. The list of elements is separated into three parts by asterisks with a category name in the center. The first category displays only one item, the name of SBS. The second one lists all the SDs, and the third category shows communication time between objects.

Component (2) contains one input field for specifying the event name and an input area to write timing constraints or time amount of communications between objects.

Component (3) displays the names of three analyzers.

Component (4) provides synchronous or asynchronous composition modes.

Component (5) lists the three analysis problems TASS can solve.

The user can input the parameters of the analysis problem in Component (6).

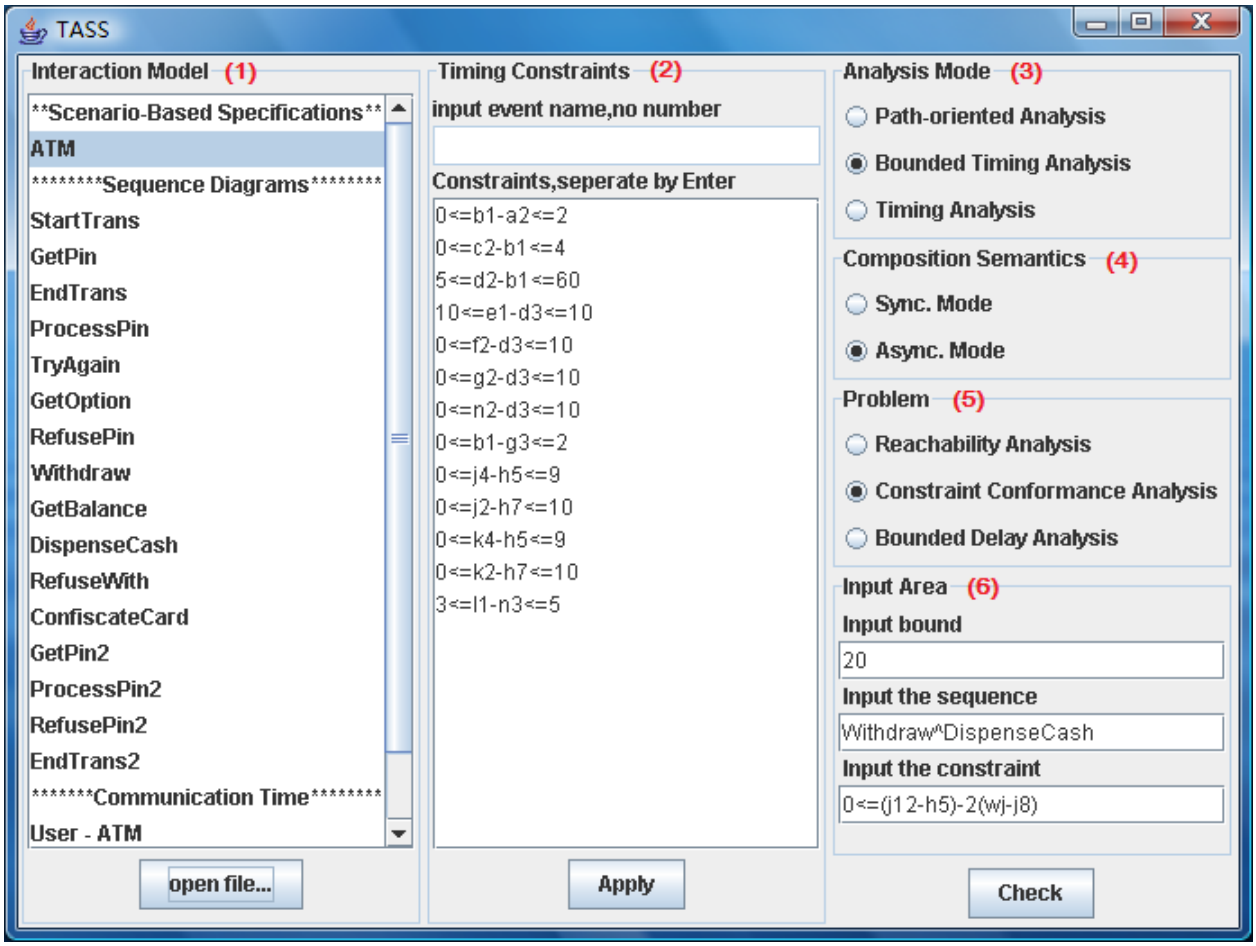

Figure 4 the TASS graphic user interface

4.5 Write the Constraints

Before write the constraints, you should first choose one item in component 1.

4.5.1 SBS Constraints

The SBS item is listed below "Scenario-Based Specifications" in component 1. When it is chosen, component 2 is not editable while component 3 becomes editable, since naming events is only applicable to SDs. Constraints are in the form of "a<=e-e'<=b". There are three points requiring special attention:

- "Less equal than" operator should always be used. That is, you should never write a constraint as "b>=e-e'>=a.
- When an equation is used, make it in the form of "a<=e-e'<=a". Obviously it is the same as e-e'=a.
- When infinites occur in a constraint, you need not (and should not) look for special character such as ∞ to represent it. You can omit the boundaries. For example, you can write "a<=e-e'<=∞" as a<=e-e'.

When all the constraints are completed, click "Apply" to save them.

4.5.2 SDs Constraints

All the SDs are listed between the items "Sequence Diagrams" and "Communication Time". When a SD is chosen, both component 2 and 3 become editable. A SD has one *event start name*. To make this concept clear, we should first know what an event is. An event is an object sending and receiving a message. To make an order for these events we need to name them. Our approach follows the rules below:

- Associate one event name to a SD. The event name can be letters, numbers, etc, which can appear in a Java String.
- Add a suffix of natural numbers starting from 1 to the event name in the visual order of the diagram. E.g. Figure 5 shows an SD with the event name "evt". This step is accomplished by TASS automatically. All user need to do is to specify the event name in Component 2.
- Each SD has an event that is temporally after all the events which indicates the end of the scenario. This special event is named following the rule that it starts with a "w", then the event name you input in the field is appended. For example, in Figure 5 the last event is wevt, which is included by TASS, and can be used while writing the constraints.

For constraints, they should appear similar to those of SBS. But one mechanism is added: parentheses are included. So the constraint is of the form: " $a<=C_0(e_0-e')+C_1(e_1-e_1')+\ldots+C_n(e_n-e_n')<=b$". Whenever there is a coefficient of the event, it should be out of the parentheses. It is meaningless for the constraint like $a<=C_0\ e-C_1e' <=b$.

4.5.3 Communication Time

It takes time for messages exchanging between objects. The time amount for every two objects exchanging messages is not the same. Yet we assume the time amount between two specified objects is fixed, so we could designate it a number. First choose one item below "Communication time". The name of the item such as "User-ATM" is named after the participating objects with a hyphen separated the two (cf. Figure 4). Then the user may input the time amount (a positive real number) in Component 2.

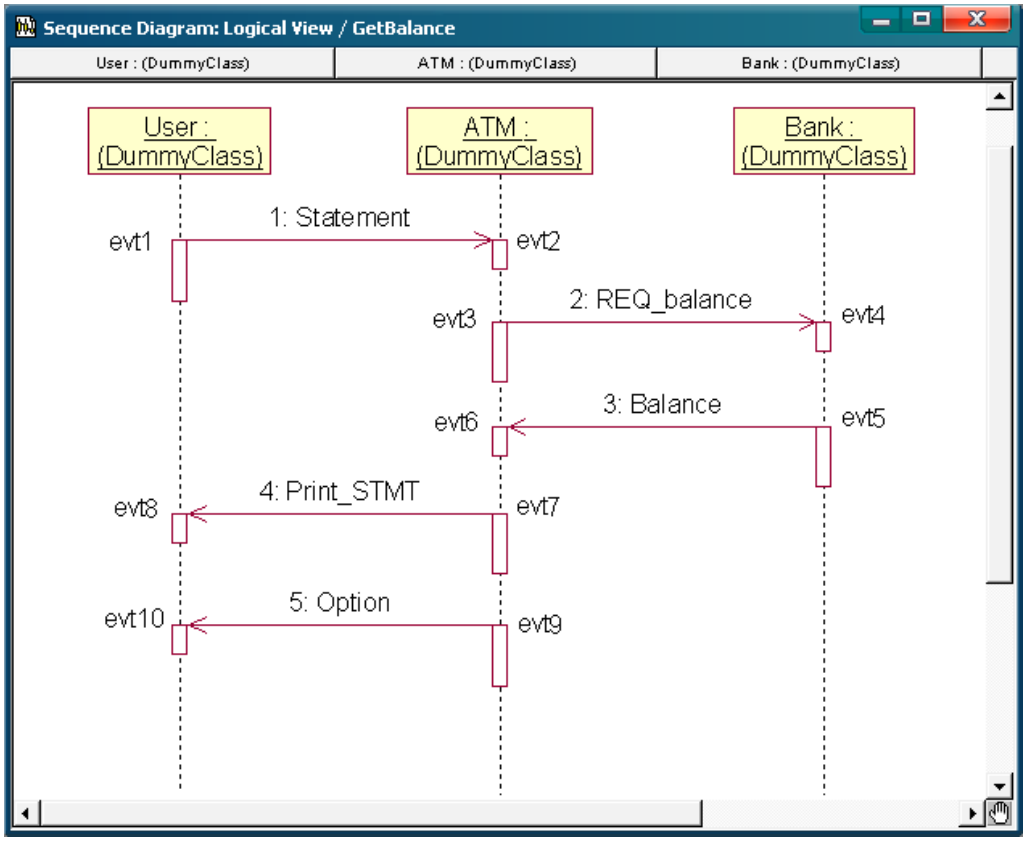

Figure 5 A SD with labeled event names

5. Timing Analysis Problems

TASS can do three types of timing analysis problems: the reachability analysis, the constraint conformance analysis, and the bounded delay analysis. We will explain them one by one. Because the catenation of two SDs in an SBS can be interpreted either as the synchronous mode or the asynchronous mode, there is an option in component 4 indicating the interpretation mode (Sync. Mode and Async. Mode).

5.1 Reachability Analysis

The reachability analysis is to check if a given node of a SBS is reachable along a behavior of the SBS. Choose the radio button “Reachability Analysis” in “Problem” panel, then put the node name in the input field which has a label saying “Input node name”. Click “Check” to begin the analyzing process.

The automatic analysis process will record all the traces and analysis results. It produces a log file named after the xml file with a suffix”. log” in the ASCII text format. The history log won’t be deleted if

new analysis is applied, for the user may consult history information during new analysis. If the file grows too big and you don't need it any more, be free to remove it from your file systems (cf. Figure 6).

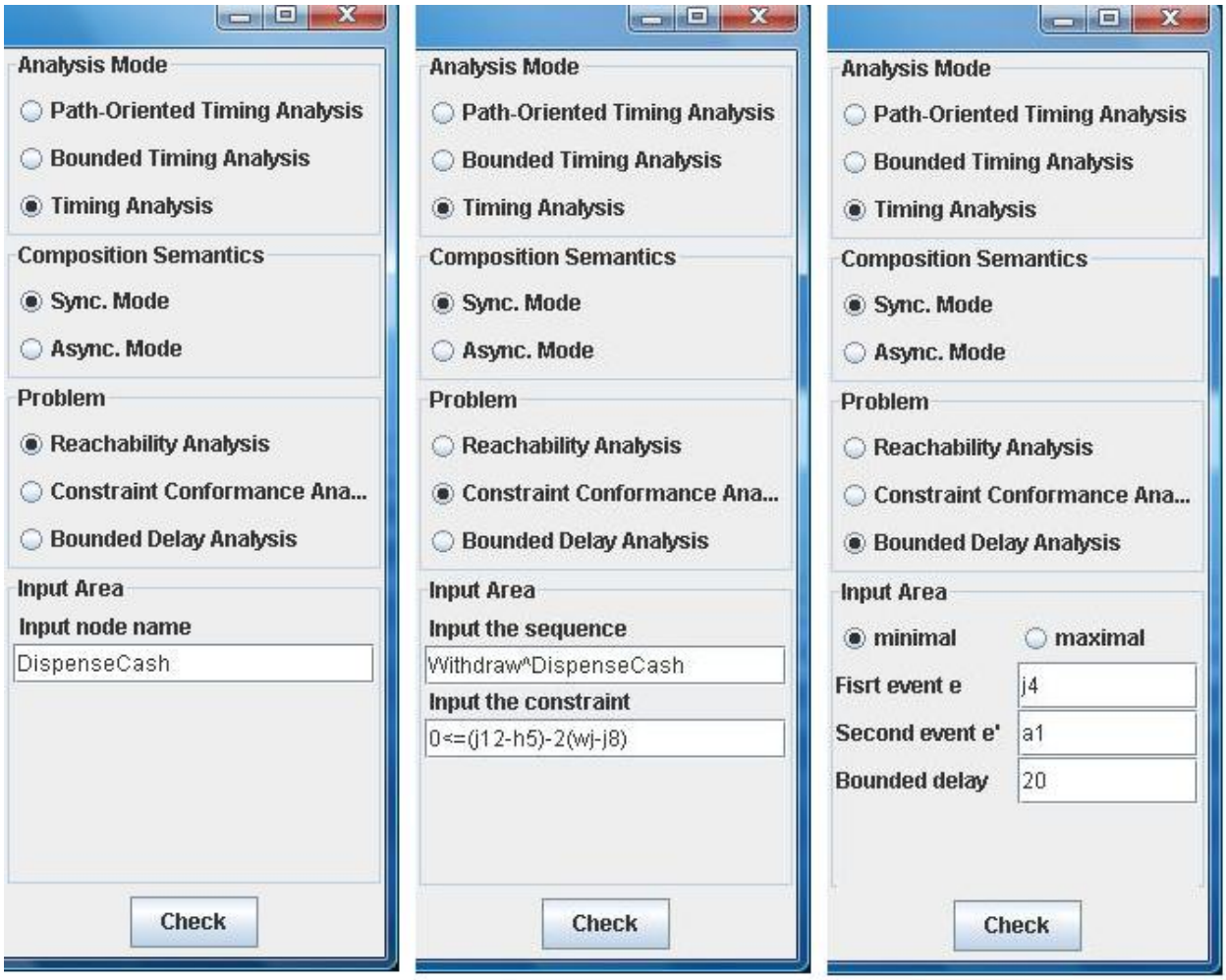

Figure 6 Options of analysis problems

5.2 Constraint Conformance Analysis

The constraint conformance analysis is to check if the given several scenarios which occur continuously in the behavior of a SBS, satisfy a given timing constraint. Choose the radio button "Constrain Conformance Analysis" in "Problem" panel. There will be two input fields. The "sequence" is a sequence of node names, with "^" to catenating them. Pay attention no blanks are permitted. The "con-

straint" is in the same form as that for a SD. E.g. Constraint like "2(wj-j8) <=j12-h5" should be rewritten to "0<=(j12-h5)-2(wj-j8). (cf. Figure 6)

The analysis process is also recorded in the log file, plus a message dialog indicating whether the constraints are satisfied or not.

5.3 Bounded Delay Analysis

The bounded delay analysis is to check if the separation in time between the two given events in any behavior of a SBS is not smaller (greater) than a given real number, which is called the minimal bounded delay analysis and the maximal bounded delay analysis respectively.

To run the analysis, choose the radio button "Bounded Delay Analysis" in "Problem" panel. Two more radio buttons labeled as "minimal" and "maximal" would appear with three input fields. The radio buttons "minimal" and "maximal" stand for the minimal bounded delay analysis and the maximal bounded delay analysis. Choose one depends on your requirements. More attention is needed for the event names. We always show the time intervals in the form e-e', so the first event e would happen after the second event e' has happened. Don't mess them up. And the last input field is for the time amount called bounded delay, which is mentioned before.

As the analysis presented above, this analysis will be logged, together with an indicative dialog.

6. Timing Analysis Mode

The timing analysis mode defines the way that TASS traverses the specification model. There are three timing analysis modes which are path-oriented timing analysis, bounded timing analysis and timing analysis.

In the path-oriented timing analysis mode we check a given path of the form $\rho \rightarrow \rho_{loop}^{k} \rightarrow \rho'$ so there are two more parameters: the path and the bound k. The path should be in the form of nodename^nodename

^(nodename^nodename)^nodename where the node names in the parentheses are ρ_{loop}. And the bound stands for the repeat times k of ρ_{loop}. The user can also choose to check a path without loops. In this case no parentheses is required the bound input field can be left blank.

In the bounded timing analysis mode we check all the potential paths of an SBS whose lengths are constrained by a threshold. So the threshold which we call "bound" should be input in the "Input Area" panel. (cf. Figure 7).

In the timing analysis mode TASS will check all the paths of SBS so no extra information about how to traverses the model is needed.

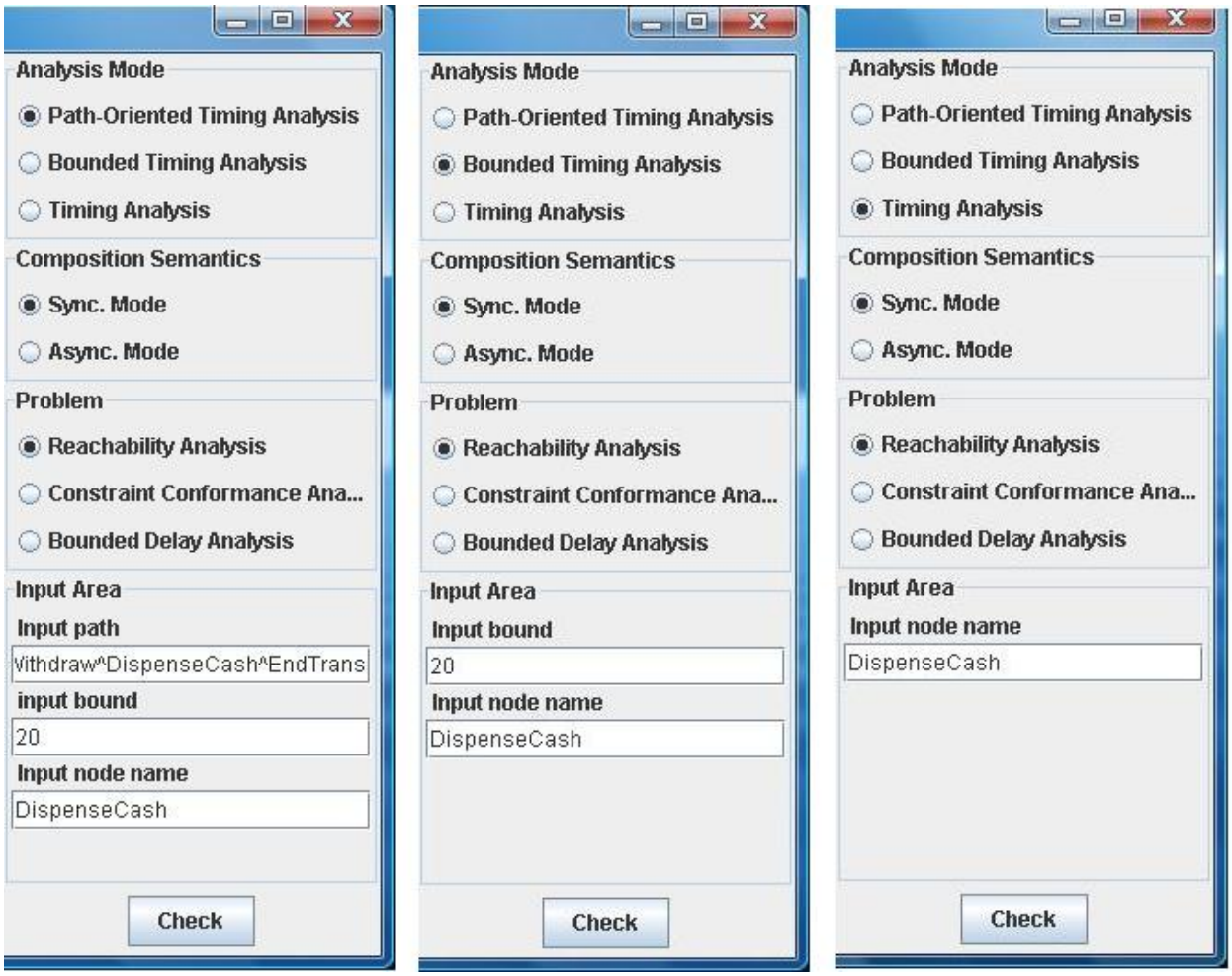

Figure 7 Options of analysis modes

Appendix A.

- IBM® Rational® Rose® Enterprise provides a common modeling language for enabling faster creation of quality software.
  Refer to: http://www-306.ibm.com/software/awdtools/developer/rose/enterprise/index.html
- XMI Add-in 1.3.6 is a Rational® Rose® Enterprise add-in which can export diagrams to XML files.
  Refer to: http://www-128.ibm.com/developerworks/rational/library/content/03July/2500/2834/Rose/rational_rose.html
- OR-Objects is a library of Java[tm] classes for developing Operations Research applications. It can develop specific solutions for linear problems as well as other ones. It is a freeware that can be downloaded at http://opsresearch.com/OR-Objects/download/free.html. TASS includes the jar file as part of its components.